Expert-level Leaf Cell Layout Generation via Preference-Optimized LLM

Yaohui Han[1], Rongliang Fu[1], Yanming Liu[2], Shuo Ren[1], Shuai Dong[2], Yunpeng Wang[2], Tinghuan Chen[2], Bei Yu[1], Tsung-Yi Ho[1]

[1] The Chinese University of Hong Kong, [2] Huawei Technologies Co., Ltd, [3] The Chinese University of Hong Kong, Shenzhen

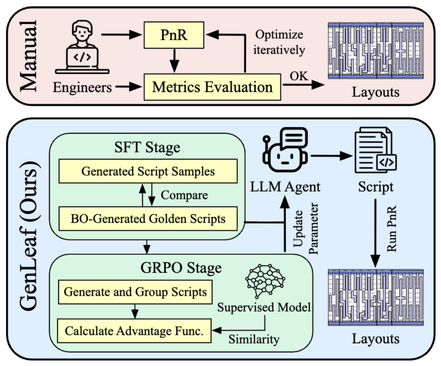
ICML International Conference On Machine Learning

1. Introduction: Leaf Cell Layout Design

Leaf cells are the basic units in the filed of integrated circuits. Each leaf cell serves as a fundamental blocks that are widely reused in various VLSI designs.

Existing Limitations:

- Still heavily depend on layout engineers.
- Without considering design experience, the results are always unreasonable or even unusable.
- Users cannot intervene the placement process through natural languages.



Comparison between our proposed GenLeaf and the traditional manual design method for leaf cell layout

Core Contributions:

- A performance-aware **representation learning** method is proposed to represent layouts.
- A BO-based **data preparation pipeline** is developed to convert expert layouts into scripts.
- A **two-stage specialization strategy** involving SFT and GRPO is proposed to allow LLM to learn tacit design knowledge.

2. Supervised Leaf Cell Layout Representation

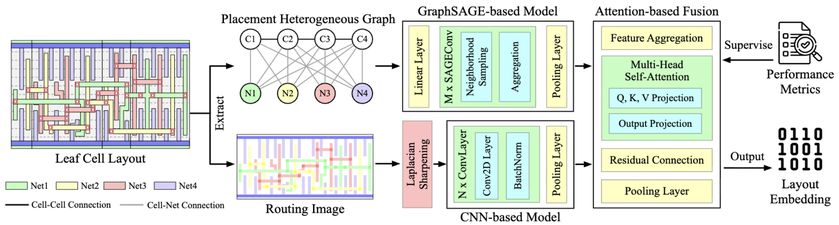
Placement: We propose a placement heterogeneous graph (PHG) to model spatial and connection relationships between cells.

Routing: Since the design scales of leaf cell layouts are significantly smaller than conventional VLSI designs, we can perform representation more detailly. The routing results are treated as images.

GenLeaf performs representation learning for two objectives: 1) Let the embeddings of cells with similar topology closer. 2) Let the embedding of cells with better performance far from those with poor performance. Therefore, we propose a supervised pipeline with the following objective:

$$\mathcal{L}_{ebd} = \sum_{(i,j)\in\mathcal{P}} \left(\mathrm{sim}(e_i, e_j) - \mathrm{sim}(\mathbf{m}_i, \mathbf{m}_j)\right)^2,$$

where **m** is a vector includes track usage, wirelength, and via count of the leaf cells.

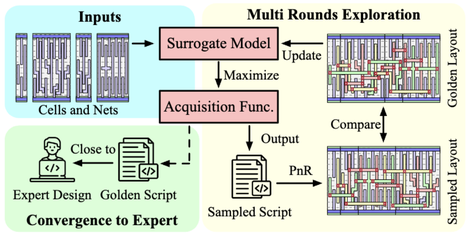


Supervised leaf cell layout representation pipeline, PnR features are processed separately.

3. BO-based Layout-Script Mapping

In physical designs, since LLM cannot fully understand the positional and rotational relationship between cells, GenLeaf formulate the layout generation task into a code generation task to enhance the overall performance.

However, all the industrial data is just layout, without any corresponding scripts or APIs. To address this problem, we utilize several PnR APIs and propose a **Bayesian Optimization** (BO)-based layout-script mapping scheme.



The BO-based layout-script mapping scheme

Objective:

$$f(x) = (\alpha \cdot t + \beta \cdot w + \gamma \cdot v) - \mathrm{sim}(e^b, e^g),$$

Metrics: Track usage, wirelength, via count

Similarity from the representation model

Variables:

$$\mathcal{O} = \prod_{i=1}^{n} \{1, 2, \ldots, n-i\},\ \mathcal{R} = \{\mathrm{R0}, \mathrm{MY}\}^n$$

Placement **permutations** after Lehmer coding

The **orientation** of cells

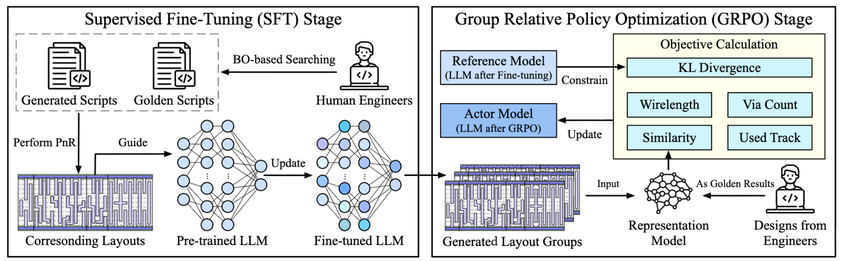
4. Specialized LLM-based Layout Generation

Since we transform the leaf cell physical design into a code generation task, it is vital to enhance the generative quality of GenLeaf. To address this problem, we utilize a two-stage specialization, including SFT and GRPO to train the LLM agent.

SFT is to enable GenLeaf to gain a basic understanding of the calling order and methods of relevant PnR APIs, and to refer to the golden scripts generated from the golden layouts.

$$\mathcal{L}_{SFT} = \frac{1}{N}\sum_{i=1}^{N}[-z_{i,y_i} + \log(\sum_{j=1}^{V}\exp(z_{i,j}))],$$

where $z_{i,j}$ is the logit of the i-th token, and z_{i,y_i} is the corresponding logit from the ground truth script.



The LLM specialization process of GenLeaf, including SFT and GRPO

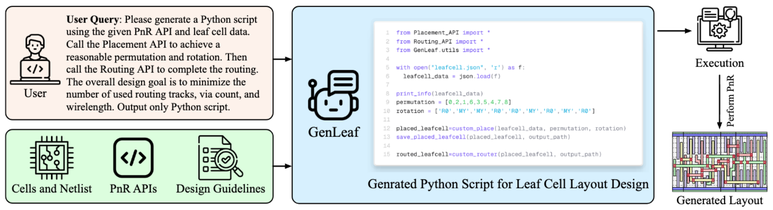
After SFT, **GRPO** is based on the fine-tuned LLM and further improves the performance by aligning with expert preferences. We customize the advantage to achieve a performance-driven policy optimization.

$$\hat{A}_j = (\alpha \cdot \frac{\mu_t - t_j}{\sigma_t} + \beta \cdot \frac{\mu_w - w_j}{\sigma_w} + \gamma \cdot \frac{\mu_v - v_j}{\sigma_v}) + \mathrm{sim}(e_j, e_j^g)$$

In the first three terms, μ and σ represent the average value and standard deviation of the three physical metrics, respectively. And the last term is the similarity between the embedding e_j of the LLM-generated layout and the corresponding golden layout's embedding e_j^g.

5. Usage of GenLeaf

The input is the user query, PnR APIs, design guidelines, and the information about the cells and netlist of the leaf cell to be designed. Then, GenLeaf inferences and generates Python scripts for leaf cell PnR design. With the help of the PnR engine, this code can be executed and output the whole leaf cell layout.



Case study of GenLeaf, which generates scripts to design leaf cell layouts

6. Experimental Results

Model	MSE	MAE
GenLeaf (Ours)	0.849	0.722
Circuit GNN (Yang et al., 2022)	1.287	0.815
DeepLayout (Zhao et al., 2025b)	1.324	0.969

Error comparison between GenLeaf and other baselines

We design a prediction task as a downstream task. The **track usage**, **via count**, and **wirelength** form a vector of elements to be predicted.

The results demonstrate the effectiveness of the representation.

Case	Golden Design			**GenLeaf (Ours)**		
	#Track	WL	#Via	#Track	WL	#Via
L1	7	29.91	18	3	30.66	18
L2	7	42.01	21	3	43.09	21
L3	8	54.03	24	3	53.72	24
L4	7	66.29	27	3	64.83	27
L5	6	29.80	25	5	32.22	22
L6	2	32.19	11	2	33.11	11
L7	13	46.22	30	7	43.71	32
L8	4	9.07	4	3	10.84	9
L9	3	49.10	8	2	36.85	8
L10	5	66.51	27	3	61.12	27
L11	9	15.41	16	5	20.89	15
L12	12	26.93	26	5	23.80	28
L13	13	62.99	31	8	62.97	23
L14	8	36.82	19	3	28.90	27
Avg. ratio	1.00	1.00	1.00	**0.52**	**0.96**	**1.01**

Performance comparison between GenLeaf and Golden Designs

We compare the performance between **Golden Designs in Huawei** and our GenLeaf.

It is notable that our GenLeaf achieved performance **close** to Golden in wirelength and Via usage and significantly **outperformed** Golden in Track usage.

The results demonstrate that GenLeaf can beat the engineers.

$$\min\left(\sum_{k=1}^{|E|}\mathrm{HPWL}_k(\mathcal{O},\mathcal{R}) + \max_{p\in\mathrm{Pin}}(\sum_{k=1}^{|E|}C_{p,k}(\mathcal{O},\mathcal{R}))\right)$$

We also compare the traditional optimization method (MILP + Channel Routing) and GenLeaf.

Method	Metric	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	Ratio
MILP + CRouter	#Track	3	3	3	3	4	3	6	3	2	4	8	5	8	3	1.00
	WL	30.83	43.75	56.67	69.60	27.48	31.72	57.09	10.18	57.16	79.12	15.85	32.20	70.87	35.34	1.00
	#Via	18	21	24	27	22	11	32	9	8	27	16	28	23	31	1.00
GenLeaf (Ours)	#Track	3	3	3	3	5	2	7	3	2	3	5	5	8	3	**0.94**
	WL	30.66	43.09	53.72	64.83	32.22	33.11	43.71	10.84	36.85	61.12	20.89	23.80	62.97	28.90	**0.88**
	#Via	18	21	24	27	22	11	32	9	8	27	15	28	23	27	**0.98**

Performance comparison of layout generation between GenLeaf and MILP + CRouter.

The reason why GenLeaf is overall superior to the traditional method is that it can learn from the design experience of experts, and explore possible better solutions through the SFT and GRPO stages.